LIFERAY

Profile

Liferay DB configuration using JNDI

« Back

General Blogs March 29, 2016 By Devang Patel

In this post I want to share liferay DB configuration using JNDI instead of portal-ext.properties.

For JDBC connection we use portal-ext.properties.

If your custom service builder portlet is pointing to another database then we can use JNDI to configure that database.

So now here I am doing this using **JNDI configuration.**

Follow below steps:

1) We have to add DB properties in server.xml instead of portal-ext.properties

Here is an example:

```
<GlobalNamingResources>
<!-- Editable user database that can also be used by
     UserDatabaseRealm to authenticate users
-->
<Resource
   name="jdbc/CustomDBPoolShared"
   auth="Container"
   type="javax.sql.DataSource"
   factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@YOUR_SERVER:1521:YOUR_SERVICE"
   username="root"
   password="root"
   maxActive="20"
   maxIdle="5"
   maxWait="10000"
/>
......
</GlobalNamingResources>
```

2) Create ResourceLink in context.xml.

```
<Context>

<!-- Default set of monitored resources -->
<WatchedResource>WEB-INF/web.xml</WatchedResource>

<!-- Uncomment this to disable session persistence across Tomcat restarts -
<!--
```

DEVANG PATEL

Blog

Recent Bloggers

Travis Cory

Staff

4 Posts

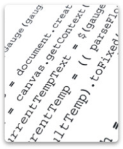
October 25, 2016

David Truong

Staff

18 Posts

October 19, 2016



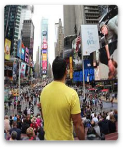
Charles Lam

2 Posts

October 17, 2016

Profile

Kyle Joseph Stiemann
Staff
7 Posts
October 17, 2016



Marcos Castro
Staff
8 Posts
October 13, 2016

Olaf Kock
Staff
118 Posts
October 12, 2016

Sven Werlen
4 Posts
October 10, 2016

Sushil Patidar
6 Posts
October 4, 2016

```
<!-- Uncomment this to enable Comet connection tacking (provides events
     on session expiration as well as webapp lifecycle) -->
<!--
<Valve className="org.apache.catalina.valves.CometConnectionManagerValve" /
-->
<ResourceLink name="jdbc/CustomDBPoolShared" global="jdbc/CustomDBPoolShare
....
</Context>
```

3) Now change ext-spring.xml of your custom portlet.

```
<bean id="customDBDataSourceTarget" class="com.liferay.portal.spring.jndi.J
    <property name="jndiName">
        <value>java:/comp/env/jdbc/CustomDBPoolShared</value>
    </property>
</bean>
<bean id="customDBDataSource" class="org.springframework.jdbc.datasource.La
    <property name="targetDataSource">
        <ref bean="customDBDataSourceTarget" />
    </property>
</bean>
```

Thats IT!

So this way we can configure liferay database configuration using JNDI.

Tweet Like 0 G+1 0

10496 Views, 2 Comments

Add Comment

Threaded Replies	Author	Date
Thanks for this post it is very clear how you...	Bob Ward	April 5, 2016 9:43 AM
JNDI is fatser than JDBC configuration. And...	Devang Patel	April 9, 2016 8:07 AM

Thanks for this post it is very clear how you achieve it. I know we have discussed this type of approach before but right now we are currently:

Profile

Maarten van Heiningen
3 Posts
September 29, 2016

Christine Reyes
Staff
2 Posts
September 27, 2016

2. in the ROOT.xml file we define the jndi like this:

```
<!--<Manager
className="com.liferay.support.tomcat.session.SessionLessManagerBase
/>-->

<Resource name="jdbc/LiferayPool"
auth="Container"
type="javax.sql.DataSource"
driverClassName="net.sourceforge.jtds.jdbc.Driver"
url="jdbc:jtds:sqlserver://cont90:1433/LR_CONT165_PROD_GA2_A"
username="liferaysqladmin"
password="L!fer@y5ql"
maxActive="500"
maxIdle="120" maxWait="20000"
removeAbandoned="true"
removeAbandonedTimeout="20"
logAbandoned="true"
maxPoolSize="100"
minPoolSize="10"
acquireIncrement="10"
validationQuery="SELECT 1"
minEvictableIdleTimeMillis="3600000"
timeBetweenEvictionRunsMillis="1800000"
numTestsPerEvictionRun="10"
testWhileIdle="true"
testOnBorrow="true"
testOnReturn="false"
/>
```

Could you comment on the advantage with your approach?

Thanks.

Sign in to vote.

Posted on 4/5/16 9:43 AM.

JNDI is fatser than JDBC configuration. And main advantage while moving or deploying code to other developement.

Patel configure each app server to use the same JNDI name, you can have different databases in each environment and not have to change your code. You just pick up the WAR file and drop it in the new environment.

Sign in to vote.

Posted on 4/9/16 8:07 AM in reply to Bob Ward.